```
//ESPECICACIONES
aux parcelasCultivo (c: Campo) : [(Z, Z)] =
[(i, j) | i ← [0..prm(dimensiones(c))), j ← [0..sgd(dimensiones(c))), contenido((i, j), c) == Cultivo] ;

Secuencia<Posicion> parcelasCultivo(const Campo &c)    //c nunca se modifica y no voy a hacer mencion a sus estados.
{
//(por asegura de dimensiones uso dimensiones.largo = sgd(dimensiones));
//(por asegura de dimensiones uso dimensiones.ancho = prm(dimensiones));
//(por asegura de contenido uso c.contenido({i,j} = contenido(c,(i,j))));
//(reemplazo push_back por ++, por ej: ts.push_back({i,j}) = ts ++ (i,j));
        Secuencia<Posicion> ts = {};
        //estado E1;
        //vale ts == {};
        int i = 0;
        //estado E2;
        //vale ts == ts@E1 ∧ i==0;
        int j = 0;
        //estado E3;
        //vale ts == ts@E2 ∧ i==i@E2 ∧ j==0;
        //Implica ts==ts@E1 ∧ i==0 ∧ j==0;
        //Vale Pc: ts == {} ∧ i==0 ∧ j==0;
        while ((j < c.dimensiones().largo) && (i < c.dimensiones().ancho)){
                //I: 0<=i<=prm(dimensiones(c)) ∧ 0<=j<=sgd(dimensiones(c))
                //   ∧ ts==[(x,y) | x <--[0..i), y<--[0..j),
                //   contenido((x,y), c)==Cultivo];
                //Fv: (sgd(dimensiones(c)+prm(dimensiones(c)))) - (i+j);
                //cota = 0;
                //B: ((j < sgd(dimensiones(c))) ∧ (i < prm(dimensiones(c))));
                //(por asegura de dimensiones uso dimensiones.largo = sgd(dimensiones));

                //Estado Ec0;
                //vale I ∧ B;
                //vale Pif1: i==i@Ec0 ∧ j==j@Ec0 ∧ ts=@ts@Ec0;
                if (c.contenido({i,j}) == Cultivo){
                        ts.push_back({i,j});
                        }
                //Estado Ec1;
                //Vale j==j@Ec0 ∧ i==i@Ec0 ∧ Qif1 ;
                //Vale Qif1: (((contenido((i@Ec0,j@Ec0), c)) == Cultivo) --> (ts == ts@Ec0 ++ contenido((i@Ec0,j@Ec0), c)));
                //                       ∨ ((contenido((i@Ec0,j@Ec0), c)) ≠ Cultivo) --> (ts == ts@Ec0));

                //Vale Pif2: j==j@Ec1 ∧ i==i@Ec1 ∧ Qif1;
                if (j == 0){
```

```
                        j = 0;
                        i++;
                }
                else {
                        j++;
                }
                //Estado Ec2;
                //Vale: (((j@Ec1 == sgd(dimensiones(c)-1)) --> ((j==0) ∧ (i == i@Ec1+1)))
                //              ∨ ((j@Ec1 ≠ sgd(dimensiones(c)-1) --> (j == j@Ec1+1) ∧ (i == i@Ec1))));
                //Implica Qif2:  (((j@Ec0 == sgd(dimensiones(c)-1)) --> ((j==0) ∧ (i == i@Ec0+1)))
                //                       ∨ ((j@Ec0 ≠ sgd(dimensiones(c)-1) --> (j == j@Ec0+1) ∧ (i == i@Ec0))));
        }
        //Estado E4;
        //Vale Qc: (i == prm(dimensiones(c)) ∧ j == sgd(dimensiones(c)) ∧ ts == parcelasCultivo(c)) (DEMOSTRACION DEBAJO);
        return ts;
        //Estado Q;
        //Vale (result == ts@E4);
        //Implica (result == parcelasCultivo(c)) (por Qc, ya que ts no se modifica despues del CICLO);
        //Implica (result == [(x, y) | x ← [0..prm(dimensiones(c))), y ← [0..sgd(dimensiones(c))), contenido((x, y), c) ==
Cultivo]);
        //(por funcion Aux de la Especificacion de parcelasCultivo(c));
}

//------------------------------------------------------------------------------------------------------------------------//
//------------------------------------------------------------------------------------------------------------------------//
## Demostraciones del TEOREMA DEL INVARIANTE ##

Pc --> I:
        Pc: ts == {} ∧ i==0 ∧ j==0;

        I: 0<=i<=prm(dimensiones(c)) ∧ (1)
           0<=j<=sgd(dimensiones(c)) ∧ (2)
           ts==[(x,y) | x <--[0..i), y<--[0..j), contenido((x,y), c)==Cultivo];(3)

(1): Como i==0 --> i>=0 y como las dimensiones del Campo deben ser siempre positivas
, entonces (i<=prm(dimensiones(c)) ∧ i>=0) --> (0<=i<=prm(dimensiones(c)));

(2): Como j==0 --> j>=0 y como las dimensiones del Campo deben ser siempre positivas
, entonces (j<=sgd(dimensiones(c)) ∧ j>=0) --> (0<=j<=sgd(dimensiones(c)));

(3): Como (i==0 ∧ j==0), podemos escribir (x <--[0..i), y<--[0..j)) como (x <--[0..0), y<--[0..0))
y [0..0) es igual a VACIO, por lo tanto  (ts==[(x,y) | x <--[0..0), y<--[0..0), contenido((x,y), c)==Cultivo]) --> (ts=={});
```

```
//------------------------------------------------------------------------------------------------------------------------//
//------------------------------------------------------------------------------------------------------------------------//

(I ∧ ¬B) --> Qc:

Qc: i == prm(dimensiones(c)) ∧ (1)
        j == sgd(dimensiones(c)) ∧ (2)
        ts == parcelasCultivo(c)); (3)

I: 0<=i<=prm(dimensiones(c)) ∧ 0<=j<=sgd(dimensiones(c))
   ∧ ts==[(x,y) | x <--[0..i), y<--[0..j),
   contenido((x,y), c)==Cultivo];

B = ((j < sgd(dimensiones(c))) ∧ (i < prm(dimensiones(c)))); --> ¬B = ((j >= sgd(dimensiones(c))) ∧ (i >= prm(dimensiones(c))));

(1): Juntando I ∧ ¬B -->(i==prm(dimensiones(c))) ∧ (2) (j==sgd(dimensiones(c)))

(3): Ahora, tomando del I: (0): (ts==[(x,y) | x <--[0..i), y<--[0..j), contenido((x,y), c)==Cultivo]),
como (i==prm(dimensiones(c))) ∧ (j==sgd(dimensiones(c))), reemplazo en (0):
(00): (ts==[(x,y) | x <--[0..prm(dimensiones(c))), y<--[0..sgd(dimensiones(c))), contenido((x,y), c)==Cultivo])
y (00) es exactamente la misma expresion de parcelasCultivo(c), entonces
 (ts==[(x,y) | x <--[0..prm(dimensiones(c))), y<--[0..sgd(dimensiones(c))), contenido((x,y), c)==Cultivo])--> ts==parcelaCultivo
(c);

 //------------------------------------------------------------------------------------------------------------------------//
 //------------------------------------------------------------------------------------------------------------------------//

 (I ∧ (Fv <= cota)) --> ¬Bc

 Fv = (sgd(dimensiones(c)+prm(dimensiones(c)))) - (i+j);

 cota = 0;

 I: 0<=i<=prm(dimensiones(c)) ∧ 0<=j<=sgd(dimensiones(c))
    ∧ ts==[(x,y) | x <--[0..i), y<--[0..j),
    contenido((x,y), c)==Cultivo];

¬Bc = j >= sgd(dimensiones(c))) ∧ (1)
      i >= prm(dimensiones(c)));  (2);

(1)∧(2): Como (Fv <= cota) --> (((sgd(dimensiones(c)+prm(dimensiones(c)))) - (i+j)) <= 0)
--> (((sgd(dimensiones(c))-j) + (prm(dimensiones(c)) - i)) <= 0) SII*
(sgd(dimensiones(c)) == j) ∧ (prm(dimensiones(c)) == i)
```

```
y del INVARIANTE tengo que (i<=prm(dimensiones(c)) ∧ j<=sgd(dimensiones(c)))--> (i==prm(dimensiones(c)) ∧ j==sgd(dimensiones
(c))).
Luego ((j >= sgd(dimensiones(c))) ∧ i >= prm(dimensiones(c)));

OBS: *SII= si y solo si...Esto se justifica ya que por el INVARIANTE, i <=prm(dimensiones(c)) y j<=sgd(dimensiones(c));
//------------------------------------------------------------------------------------------------------------------------------//
//------------------------------------------------------------------------------------------------------------------------------ //
```